REMARKS

This Amendment is intended to be a complete response to the Office Action of May 3, 2000, and the application is believed to be in condition for allowance. Accordingly, reconsideration is respectfully requested.

Amendments to Specification:

In Office Action, the Examiner indicated that the amendment to page 5, line 12 was not entered because there are two locations of the word "a" in that line. Applicant has clarified the amendment here to avoid that ambiguity. Entry of the amendment is respectfully requested.

Status of the Claims

Claims 1-58, 91-112,114-122, and 124-136 were rejected in the Office Action. Claims 113 and 123 were objected to in the Office Action. Claims 113 and 123 are allowable. Claims 1, 29, 31, 91, 93, 95, 98, 105, 106, and 120 are amended herein. Claims 101 through 104 are cancelled herein. New Claims 137 through 144 are added in this communication. After the entry of these amendments and cancellations, Claims 1-58, 91-100, and 105-144 are pending in the application.

The Claims

35 USC 102 and 35 USC 103

Claims 1-28, 31-58, 91-97, and 100-105 were rejected under 35 USC 102(e) as being anticipated by Peyret et al. (U.S. Patent Number 5,923,884, hereafter "Peyret" or the "'884 patent") and Claims 29, 30, 97, 98, 114, and 129 were rejected under 35 USC 103(a) as being unpatentable over Peyret as applied to claims 1 and 3, and further in view of Martineau (U.S. Patent Number 5,915,226). Applicants traverse these rejections.

Applicants thank the Examiner for indicating that Claims 113 and 123 are allowable.

Claims 1, 31, 95, and 105

Applicants have amended Claims 1, 31, 95, and 105 to incorporate the limitations of allowable claims 113 and 123. For example, Claim 1 has been amended to recite the following limitations:

> "a memory storing:
>
> an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including at least one step selected from a group consisting of
>
> recording all jumps and their destinations in the original byte codes;
>
> converting specific byte codes into equivalent generic byte codes or vice-versa;
>
> modifying byte code operands from references using identifying strings to references using unique identifiers; and
>
> renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and
>
> an interpreter operable to interpret such an application derived from a program written in a high level programming language format."

Since these claims have been amended to include similar limitations as those of allowed Claims 113 and 123, Applicants respectfully submit that the reasons the Examiner applied to allow 113 and 123 are applicable to these claims. Applicants further note that neither Peyret nor the other prior art teaches or suggest these limitations.

Accordingly, Applicants respectfully submit that Claims 1, 31, 95, and 105 recite allowable subject matter and should be allowed.

Claim 127

With respect to Claim 127, the Examiner stated "see the rejections of claims 124-126, the abstract, figures 3, and 8-9." Applicants respectfully submit that Claim 127 is more closely related to Claim 113 and 123 and should be allowable for the same reasons that those claims were held allowable. Accordingly, Applicants respectfully request withdrawal and allowance of Claim 127.

Claims 91, 93, 106 and 120

Claim 91 recites "a memory storing a first application that has been processed from a second application having a plurality of language elements including at least one string of characters, the string of characters being replaced in the first application by an identifier" and Claim 93 recites "replacing the string of characters of the first application with an identifier in the second application".

In the Office Action, the Examiner asserted that "the features of claims 91, 93, and 105 are taught via claim 1." Claim 1 in the form it existed prior to this response did not recite the limitation of "the string of characters being replaced in the first application by an identifier". Therefore, any rejection with respect to Claim 1 is not applicable to Claims 91 and 93. Applicants submit that the prior art does not teach or suggest these limitations from Claims 91 and 93.

Claim 106 recites "A microcontroller having a set of resource constraints". To enable the execution of high level language programs, the microcontroller of Claim 106 recites that the applications to be interpreted are generated by a two-step process, namely:

> "a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and
>
> b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter."

Claim 120 recites "compiling the application program in the first programming language into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine;

converting the first intermediate code into a second intermediate code; wherein the second intermediate code is interpretable within the set of resource constraints by at least one second intermediate code virtual machine."

In other words, the application programs are compiled and then converted. The compilation may be, for example, a standard compilation process to generate an intermediate code, e.g., JAVA byte codes.

With respect to Claims 106 and 120 the Examiner stated "it is considered an inherent feature for a system to have means for including required attributes, producing output in a form suitable for interpretation (i.e. Bytecode for Java), converting means." Applicants disagree. The conversion process in the claimed invention reduces the intermediate code to "a minimized form suitable for interpretation within a set of resource constraints." Applicants are not aware of any prior art system that included such a conversion step. Applicants respectfully submit that the conversion process, is a post processing step not found in prior art systems including Peyret.

Legal standard for anticipation

Single reference must teach each element of the claimed invention

It is well-established law that a reference does not anticipate a claimed invention unless each element of the claim is taught by the reference. *See e.g.,* General Electric Co. v. Nintendo Co. (CA FC) 50 USPQ2d 1910 ("A judgment of invalidity for anticipation requires that a single prior art reference disclose every limitation in a patent claim."); PPG Indus., Inc. v. Guardian Indus. Corp., 75 F.3d 1558, 1566, 37 USPQ2d 1618, 1624 (Fed. Cir. 1996) ("To anticipate a claim, a reference must disclose every element of the challenged claim and enable one skilled in the art to make the anticipating subject matter.").

Legal standard for obviousness

Examiner must establish Prima Facie case of obviousness

To reject a claim under 35 USC 103 as unpatentable over a reference, the Examiner must set forth a case of prima facie obviousness. In re Fine (CA FC) 5 USPQ2d 1596 ("The PTO has the burden under section 103 to establish a prima facie case of obviousness."). Of course, since the Examiner rejected the independent claims as anticipated under 35 USC 102(e), the Examiner has not made a determination of non-obviousness of the independent claims. Nevertheless, Applicants respectfully submit that a prima facie case of obviousness cannot be made based on the prior art of record.

Invention must be suggested by prior art

"Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under section 103, teachings of references can be combined only if there is some suggestion or incentive to do so." In re Fritch (CA FC) 23 USPQ2d 1780 (8/11/1992) "The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification." In re Fritch at 1783.

Application of legal standard to issue of whether invention is anticipated or obvious over Peyret

Peyret does not teach or suggest Applicants' novel and non-obvious invention of how programs written in JAVA or other high level programming languages may be executed on a smart card.

The Examiner may have misunderstood Applicants' previously made argument (Correspondence of February 3, 2000). Applicants argue there that because Peyret does not teach or suggest the claimed invention, Applicants' invention is not anticipated by nor obvious over Peyret. To strengthen that argument, Applicants demonstrated that Peyret does not teach anything about Java or any other high level programming language on a smart card. Because Peyret lacks a teaching of how to use high level programming languages for programming smart cards, it follows that Peyret must fail to teach applicants novel and non-obvious invention of how to do so. Even if the Examiner's

premise that Peyret does teach something related to Java were accepted, it does not follow that Peyret teaches or suggests Applicants' novel and non-obvious invention of how to accomplish the use of high level languages on smart cards or other microcontrollers.

The Examiner cites the sentence "the applets run through the interpreter so that the applets do not have any direct access to the hardware of the smart card", Peyret, col. 5 lines 44-46 to indicate that Peyret pertains to JAVA. As discussed in Applicants' response of February 3, 2000 (incorporated herein by reference), neither the word "applets" nor "interpreter" may be considered a teaching of the use JAVA or any other high level programming language. *See for example,* U.S. Patent No. 5,548,745, to Egan et al., filed January 19, 1993 which uses the term "applet" in a distinctly non-Java context.

The Examiner has made the assertion that "the features taught by Peyret reads directly on the Java programming language" (Office Action, page 2). The fact that Peyret uses words that are also frequently associated with Java does not lead to the conclusion that Peyret had Java in mind. All squares are rectangles, but not all rectangles are squares. All Cokes have bubbles, but not all sparkling drinks are Coke. Java programs are often called *applets*, but not all *applets* are Java applets. Java programs are interpreted, but not all interpreters interpret Java. Peyret uses the words *applet* and *interpreter*, however it does not follow that Peyret had Java in mind.

It would be clear to a person skilled in the art at the time of Peyret's patent application that Peyret had in mind something other than Java because at that time, while there were multi-application smart cards in existence, none of those smart cards executed high level languages. Peyret does not deal with a solution of how to squeeze Java or another high level language onto a smart card (Peyret discloses the loading of programs onto a smart card, but not the language these are written in), so the only logical inference that can be made is that Peyret used as his starting point the technology that he had available.

Patrice Peyret has recently indirectly acknowledged that Applicants were the first to use Java on a smart card. Applicants quote from a recent publication by one of Mr. Peyret's co-workers:

> "The Java Card APIs were first introduced in November 1996 by a group of engineers in Schlumberger's product center in Austin, Texas, working to bring smart card development into the mainstream while preserving smart card security. They soon recognized that the Java programming language was the solution. Schlumberger proposed the initial draft for the Java Card APIs and became the first licensed smart card company. ...
>
> ... [Sun's] first move was to acquire Integrity Arts, a spinoff of Gemplus that specialized in the development of virtual machine and operating system technologies for smart cards." Zhiqun Chen, *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*, page 9 (2000) (Copy enclosed).

Mr. Peyret wrote the foreword to this book and undoubtedly reviewed the manuscript. He would have objected to the above quote had it not been correct. As stated in the quoted passage, Integrity Arts was in the field of virtual machines for smart cards, Mr. Peyret worked for Integrity Arts at the time he filed his patent application, and Schlumberger was first to introduce Java Card APIs, therefore, it follows that Mr. Peyret was not referring to Java in the '884 patent even if there is some similarity in terminology. Furthermore, a person of ordinary skill in the art of smart cards studying the '884 patent at the time that Applicants filed their application would not have inferred that the '884 patent was referring to Java or any other high level programming language. Therefore, the '884 patent would not have suggested to a person of ordinary skill in the art how to use a high level language on a smart card.

Nevertheless, even if, for the sake of argument, one accepts the Examiner's premise that Peyret had in mind executing programs written in a high level language on a smart card, Peyret does not teach or suggest how to accomplish that feat. Consider the notion of inter-stellar space travel and popular science fiction literature and motion pictures such as *Star Trek*. Certainly *Star Trek* does not teach or suggest *how* to travel to

distant galaxies and should a person invent a method or machine to accomplish such travel, *Star Trek*, even with its concepts of warp drives, would not render that invention anticipated or obvious. Similarly, even if the notion of writing a high level language program and to have it execute on a smart card had occurred to Mr. Peyret, he did not teach or suggest how to do that in his patent, and therefore, to the person of ordinary skill in the art studying that patent (Peyret) Applicants' invention would not be obvious.

Applicants have invented how to achieve that very difficult task. Making it possible to run programs written in a high level language on a smart card was not obvious prior to Applicants' invention thereof should be beyond dispute. To put Java (or any other high level language) on an integrated circuit card is anything but obvious. At the time of the invention, the typical Java Virtual Machine required over 1 MB of memory. Any person of ordinary skill would realize that to squeeze such an interpreter into an integrated circuit card (such as a smart card) is anything but an obvious task. Applicants direct the Examiner to the following statement made about the author of a recent JAVA WORLD article (http://www.javaworld.com/javaworld/jw-04-1998/jw-04-ringfever.html):

> "About the author: Chuck McManis wrote the very first Crypto toolkit in Java when he was a member of the Java development group in 1993. **He also brought in the first smart cards (Hitachi) at FirstPerson but gave up trying to put Java on them**. (Now he knows better!) Today he writes columns and articles for JavaWorld and is currently holding the position of director of systems software at FreeGate Corporation in Sunnyvale."

Applicants submit that Chuck McManis is well beyond a person of ordinary skill in the art of Java programming. In fact, he is a leading expert in the field of software design and holds at least four patents for his work. The fact that he "gave up trying to put Java on [smart cards]" is indicative of the non-obviousness of doing so.

Applicants also invite the Examiner to consider the following excerpt including a quote from Scott McNealy (CEO of Sun Microsystems, the company that originally developed the Java language):

"Like Playing Golf in a Phone Booth"

A smart card enabled by Java Card technology, dubbed the "world's thinnest computer," places the Java platform inside the .8 millimeter thickness of the plastic card, thus making it a small but integral part of a larger enterprise information system.

Its creation was no small feat. "Fitting Java technology inside smart cards was like playing golf inside a telephone booth," remarks Sun CEO Scott McNealy.

http://softwarema.usec.sun.com/features/1999/01/javacard.html
JAVA CARDTM TECHNOLOGY GROWS UP SMART
by Janice J. Heiss and John Papageorge

When two *experts* in the field of the Java programming language consider putting Java on smart cards very difficult, for a person of *ordinary* skill it would certainly be unattainable. Peyret's disclosure neither mentions nor suggests high level languages nor how to put such language interpreters on a smart card. Just as *Star Trek* would not enable the ordinarily skilled person to achieve inter-stellar travel, Peyret does not enable the ordinarily skilled person desiring to run high level language programs on smart cards.

Specific discussion of the independent claims 91, 93, 106 and 120

Claim 91 recites "a memory storing a first application that has been processed from a second application having a plurality of language elements including at least one string of characters, the string of characters being replaced in the first application by an identifier" and Claim 93 recites "replacing the string of characters of the first application with an identifier in the second application".

More importantly, as discussed above, Peyret does not teach or suggest how to execute programs written in high level languages on a smart card. It is therefore not surprising that Peyret does not teach or suggest replacing strings of characters with identifiers. Accordingly, Claims 91 and 93 are neither anticipated by Peyret nor obvious over Peyret and should be allowed.

As noted above, Claim 106 recites "an interpreter loaded in memory and operable within the set of resource constraints," and "at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programming environment comprising: a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter."

Similarly, Claim 120 recites "into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine" and "converting the first intermediate code into a second intermediate code; wherein the second intermediate code is interpretable within the set of resource constraints by at least one second intermediate code virtual machine".

Considering that Peyret does not teach or suggest how to achieve execution of high level language applications on a microcontroller, it is not surprising that Peyret does not teach or suggest Applicants' novel and non-obvious invention that includes an interpreter that is operable to interpret applications generated from such a two-step process or the two-step process itself.

In the Office Action the Examiner stated with respect to Claims 106 and 120 that "It is considered an inherent feature for a system to have means for including required attributes, producing output in a form suitable for interpretation (i.e., Bytecode for Java), converting means". Claims 106 and 120 recite that the compiled form is converted into a form suitable for interpretation within a set of resource constraints. A typical JAVA virtual machine interprets the byte codes from the compiler directly. Therefore, the conversion step is not inherent in a Java system.

For the foregoing reasons, Claims 106 and 120 are not anticipated by Peyret or any other known prior art and are patentable and should be allowed.

The Examiner rejected Claims 29, 30, 97, 98, 114, and 129 under 35 USC 103(a) as unpatentable over Peyret in view of U.S. Patent No. 5,9144,226 to Martineau (hereinafter Martineau). The Examiner relied on Martineau for a teaching of providing communications to a remote location. Martineau does not teach or suggest "a memory storing: an application derived from a program written in a high level programming language format, and an interpreter operable to interpret such an application derived from a program written in a high level programming language format" (Claim 1). Therefore, Claim 1 is patentable over Martineau, whether taken singly or in combination with Peyret and should be allowed.

The argument above with respect to Claim 1 is applicable to all other independent claims in the application, each of which recite elements not taught or suggested by Martineau. For these reasons, all the independent claims are patentable over Peyret and Martineau taken singly or in combination. Accordingly, the independent claims should all be allowed.

New Claims

Claims 137 through 144 are added herein. Applicants respectfully submit that the above arguments in support of the independent claims already in the case are applicable to the new claims. Accordingly, Applicants respectfully requests consideration and allowance of these new claims.

Dependent Claims

The dependent claims in the application all depend from independent claims that have above been demonstrated to be patentable. These dependent claims incorporate all the limitations of their respective base claims, recite further unique and non-obvious combinations, and are patentable for the same reasons given in support of the various independent claims from which they derive and by virtue of such further combinations.

Conclusion

It is submitted that all the claims now in the application are allowable. Applicants respectfully request reconsideration of the application and claims and its early allowance. If the Examiner believes that the prosecution of the application would be facilitated by a telephonic interview, Applicants invite the Examiner to contact the undersigned at the number given below.

It is believed that no additional fees are due in connection with this Response as has been indicated on the transmittal letter. If Applicant is in error as to these fees, the Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account 19-0597.

Respectfully Submitted,

Pehr B. Jansson
Registration No. 35,759

Date: 8/21/2000

Enclosures:
1. Acknowledgment Postcard
2. Transmittal Form (1 sheet)
3. Amendment Transmittal Letter and duplicate copy (2 sheets)
4. Pages from Zhiqun Chen, *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*, (2000) (7 sheets)

Pehr B. Jansson
Schlumberger Austin Technology Center
8311 North FM 620
Austin, TX 78726
Tel: 512-331-3748
Fax: 512-331-3060